Computational Intelligence
An International Journal

Computational Intelligence, Volume 28, Number 2, 2012

PROBABILISTIC HEURISTICS FOR HIERARCHICAL WEB DATA CLUSTERING

Morteza Haghir Chehreghani, Mostafa Haghir Chehreghani, and Hassan Abolhassani

Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Clustering Web data is one important technique for extracting knowledge from the Web. In this paper, a novel method is presented to facilitate the clustering. The method determines the appropriate number of clusters and provides suitable representatives for each cluster by inference from a Bayesian network. Furthermore, by means of the Bayesian network, the contents of the Web pages are converted into vectors of lower dimensions. The method is also extended for hierarchical clustering, and a useful heuristic is developed to select a good hierarchy. The experimental results show that the clusters produced benefit from high quality.

Recieved 29 December 2008; Revised 3 August 2010; Accepted 7 August 2010; Published online 23 April 2012

Key words: data mining, Web clustering, Bayesian networks, hierarchical clustering, representative point.

1. INTRODUCTION

Clustering data is an important task in information processing. It can be performed as a preprocessing phase to some other tasks (e.g., classification) or be used to structure data for better visualization and use by humans. On the Web, this task has an important role: more than a preprocessing step, it can be also used for enhancing search engine results, facilitating Web crawling and organizing domain knowledge.

So far, various methods have been developed for clustering Web data, employing different measures such as link structure, content, and combinations of both (Wang and Kitsuregawa 2001, 2002; Getoor 2003). In general, the Web clustering algorithms can be divided into two main categories: hierarchical algorithms and partitioning algorithms. Hierarchical algorithms represent data in multilevel tree structures, while partitioning methods try to partition data into separate, dissimilar clusters (Jain, Murty, and Flynn 1999; Xu and Wunsch 2005). Our work in this paper is in the field of hierarchical clustering.

Recently, some novel concepts and techniques have been introduced in Web data mining, including some borrowed from the machine learning context (Stumme, Hotho, and Berendt 2001, 2006; Zhang and Lee 2004). Generally, these techniques learn and extract more pure knowledge from dirty data (especially text data) and provide some level of supervision to the unsupervised clustering process. Some of them first extract knowledge from the domain and then cluster the data associated to that domain. Aligned with this approach, in this paper, we introduce a new clustering method based on learning a domain, with the aim of extracting more accurate representatives for the cluster centers. Such knowledge is extracted by inference from a Bayesian network. Moreover, the method is extended to construct hierarchical clusters. Evaluation of the experimental results show considerable improvements in the clustering quality.

The remaining the paper is organized as follows: in Section 2 we present a brief overview of related works. In Section 3, we first introduce the steps of constructing a domain vocabulary and associated Bayesian network, and then we explain the process of grouping the categories

Address correspondence to Morteza Haghir Chehreghani, Faculty of Computer Engineering, Web Intelligence Laboratory, Department of Computer Engineering, Sharif University of Technology, Azadi Avenue, Tehran, P.O. Box: 11365-11155, Iran; e-mail: haghir@ce.sharif.edu

via an efficient algorithm and assigning the Web pages to cluster centers. In Section 4, we extend the proposed algorithm to provide hierarchical clustering, and also in that section we prove some theorems that will be useful in selecting better hierarchies. In Section 5, the experimental results are evaluated and a new measure for the evaluation of hierarchical clusters is developed. Finally, the paper is concluded in Section 6.

2. RELATED WORKS

Clustering is a classic data mining problem and many different methods have been developed to address it. Some of the well-known clustering methods include partitioning algorithms based on dividing entire data into dissimilar groups, hierarchical methods, density- and grid-based clustering approaches, and graph-based methods (Barthelemy and Leclerc 1995; Jain et al. 1999; Grira, Crucianu, and Boujemaa 2005; Xu and Wunsch 2005). Due to the high dimensionality and lack of orthogonality of Web document vectors, algorithms such as K-Means (McQueen 1967), hierarchical agglomerative clustering (HAC) (Day and Edelsbrunner 1985), and graph partitioning methods have gained more popularity and applicability in the Web environment (Yao and Choi 2003).

The initial research on clustering Web content was based on text mining and traditional information retrieval techniques. They mostly use vector models of the contents of Web pages and neglect their hyperlink structure. Later, several methods were developed to take advantage of the Web structure, and especially the link analysis (Wang and Kitsuregawa 2001). The next wave of efforts inclined toward using a combination of link structure and content information. Thus according to the clustering criteria, Web clustering algorithms can be divided to three categories: link-based methods, content-based methods, and combinations of both (Wang and Kitsuregawa 2002; Getoor 2003; Huang and Lai 2003).

HAC methods are popular clustering methods that combine smaller data items to create bigger clusters. An HAC algorithm starts with each Web page in a single cluster and merges the two most similar clusters iteratively until a halting criterion is satisfied. The measure used for combination can be the nearest, the average, and the farthest distances, and these methods can thus be classified into several categories (Day and Edelsbrunner 1985): single linkage, average linkage, and complete linkage.

(1) One of the most common clustering algorithms is K-Means. It has several advantages, such as its simplicity and speed of convergence. Nevertheless, this algorithm has some drawbacks including the problems of selecting appropriate initial centroids, convergence to a local optimum, and sensitivity to noisy and outlier data (Xu and Wunsch 2005). In fact, K-Means restricts the search space to a limited subspace in the vicinity of the initial centers until it achieves a local optimum. To solve these problems and adapt the algorithm to the properties of the Web environment, several enhancements have been developed. The following three main categories of improvements have been developed.[1]
(2) Methods that try to perform several runnings of K-Means whether on the same data set or on different partitions of the whole data set. These methods were mostly developed for parallel clustering and a useful review can be found in Bandyopadhyay et al. (2006). Of course, sequential (nonparallel) versions of these methods were also developed. A refinement of K-Means is presented in Bradley and Fayyad (1998) that first applies the K-Means algorithm M times to M random subsets of the original data. Then the union

[1] Of course, other improvements have also been developed. For example, in Wang and Kitsuregawa (2001), a measure is used for finding the cluster centers and updating them that also considers the links.

of the solutions of these subsets is formed and clustered M times again, setting each subset solution as the initial guess. The initial points for the whole data are obtained by choosing the solution with the minimal sum of squared distances. A global K-Means algorithm, including a series of K-Means procedures with the number of clusters varying from 1 to K, is presented in Likas, Vlassis, and Verbeek (2003). In this algorithm, at each k, $k = 2, \ldots, K$, the previous $k-1$ centroids are fixed and the new centroid is selected by examining all the data points. Nevertheless, this solution has the problem of computational complexity, making it most likely to be useful for parallel computations. Another technique called ISODATA was also developed in Ball and Hall (1967), which adjusts the number of clusters by merging and splitting operations repeatedly according to some predefined thresholds.

(3) Methods that try to explore the search space more deeply and usually use different jump operators to examine more areas of the search space. Therefore, these methods are not so sensitive to the initialization. Most of the optimization techniques fall into this category. The first attempts at using the optimization techniques in the clustering area were initiated by Genetic algorithms, as explained in Raghavan and Birchand (1979). Other attempts can be found in Gareth et al. (1995) and Krishna and Murty (1999). The enhanced LGB algorithm of Patane and Russo (2001) adopts a roulette mechanism typical of genetic algorithms to find near optimal results. However, other optimization techniques, such as Ant clustering (Labroche, Monmarché, and Venturini 2003) and Particle Swarm Optimization (PSO; Kennedy, Eberhart, and Shi 2001) on image and low-dimensional data sets in Merwe and Engelbrecht (2003) and Omran, Salman, and Engelbrecht (2002), and for document clustering in Cui, Potok, and Palathingal (2005), have been used to improve the clustering results. Moreover, Abidi and Ong (2000) combine the K-Means with self-organizing neural networks. Finally, in Xing et al. (2003), K-Means is applied using the Mahalanobis distances adjusted by convex optimization, and an adaptive learning rate strategy for on line K-Means is developed in Chinrungrueng and Séquin (1995) .

(4) The last category contains the methods that first extract some knowledge in a preselected domain. This knowledge is used to provide useful information about the cluster representatives and even their numbers. Both of the previously described categories suffer from high computational costs; although these methods spent an initial cost to extract the knowledge, they can however then complete the later processes with lower computational cost. Because our work lies in this category, we now focus on some of the recent methods developed using this approach.

Some recent trends in the extension of semantic Web applications are inclined to use semantic Web concepts in Web mining tasks (Stumme et al. 2001, 2006; Zhang and Lee 2004). In the semantic Web, *Ontology* and *Logic* have basic roles and can be used as powerful tools for the enrichment of Web documents. As defined in Gruber (1993), Ontology is "an explicit formalization of a shared understanding of a conceptualization." This definition and its more precise expressions in tasks (Stumme et al. 2001, 2006) suggest a good idea for mining the data in a specific domain: we can construct an Ontology for the domain that contains common concepts, in the hope that this will help solve the problems such as high dimensionality. However, extracting an Ontology from a Web domain is a challenging problem in and of itself. One way is to create the Ontology by hand, which is expensive. Therefore, the concept of Ontology learning has been developed. In Bozsak et al. (2002) and Maedche and Staab (2001), machine learning techniques have been used to improve the Ontology engineering process and to reduce the cost of constructing an Ontology. After constructing the Ontology, we have only a limited number of concepts. Then, as in the classification (Koller and Sahami 1997; Bloehdorn and Hotho 2004), different mining operations can be done with higher

Algorithm 1 (Steps of the proposed method)**:**

1. Find the **concepts** of the domain.
2. Construct the Bayesian Belief Network including concepts and complete it by initial conditional probabilities.
3. Identify the **categories** in the network.
4. Construct the vectors of the categories.
5. Group the categories and consider them as cluster centers.
6. Construct the vectors of the web documents of the test collection (by categories).
7. Assign the document vectors to the cluster centers.

FIGURE 1. Steps of the proposed method based on finding representative points.

quality on these limited concepts. In Hotho, Maedche, and Staab (2001) and Hotho, Staab, and Stumme (2003), Ontologies have been used as the background knowledge during the preprocessing step and Ontology-based heuristics have been applied to the input data for feature selection and aggregation. A similar study has been done on classification in Bloehdorn and Hotho (2004). Other instances of using semantic Web technologies include text mining and categorization (Sebastiani 2002), Web service annotation (Hess and Kushmerick 2004), learning content categories from usage by hidden Markov models (Ypma and Heskes 2002), and so on.

In spite of these improvements, Ontology contains some limited relationships and cannot infer the relationship between each pair of concepts extracted from the domain. In the semantic Web, this task is delivered to the logic layer. All reasoning and logical inferences are done at this layer. Unlike Ontologies, there is not much existing research on using logic in the context of Web mining in general and of data clustering in particular. As will be investigated in this paper, an alternative approach that can provide the inference between the concepts of a specific domain is to use Bayesian Belief networks (Hecherman 1995; Mitchell 1995). Using this technique enables us to extract the hidden relationships that are not stated explicitly by the document vectors.

3. FLAT CLUSTERING USING BAYESIAN NETWORKS

The proposed algorithm uses the Bayesian network to determine the representative points of the *K-Means* algorithm. To do so, first, a specific domain, such as *Politics,* is identified, and then a *vocabulary* (set of *concepts*) is extracted for it. Then this vocabulary is converted into a Bayesian network, which can infer and extract relationships between any two concepts. The concepts that contain more general semantics are selected as *categories,* which constitute the main dimensions of document vectors (of the test data set). In the next step, these categories are grouped according to the relationships between them. The grouping (similar to clustering) can be done either in hierarchical or in flat form. Each group corresponds to a center point of the *K-Means* algorithm.

Therefore, this approach yields the proper number of clusters as well as the appropriate representative point of each one. Then, each Web page of the test collection is converted into a vector with dimension equal to the number of categories. Finally depending on the similarity between each Web page and each of the centers, the Web page is assigned to the most suitable center. These steps are shown in Figure 1. In the following, we explain each step in detail.

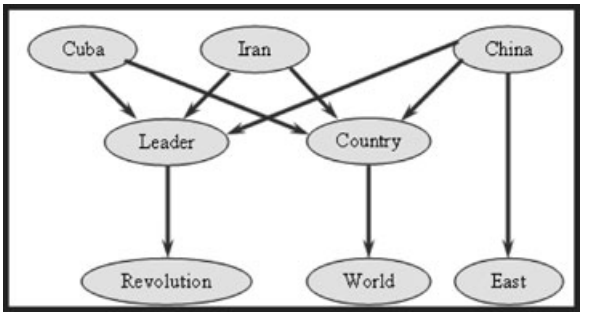

FIGURE 2. An example of organizing concepts in the Bayesian network: concepts are partially ordered from more specific concepts to more general concepts.

It is important to note that in the following explanations, we introduce and use some parameters and thresholds, as is common in other clustering methods that do not rely on direct (pairwise) similarities between objects. For example, this issue is clear in the method proposed in Steinbach, Ertoz, and Kumar (2003), which requires five parameters that must be set empirically (*strong link threshold*, *noise threshold*, *topic threshold*, *merge threshold*, and *labeling threshold*). Most of the parameters in our method are related to the training (knowledge extraction) stage, and they can be estimated by some tuning.

3.1. Constructing the Bayesian Belief Network and Finding Categories Inside the Network

There exist several manual, semiautomatic, and automatic methods to construct a vocabulary. In our approach, first a training set is selected, and after performing conventional preprocessing such as the elimination of stopwords and stemming, the remaining terms are sorted according to the *term frequency* $\times$ *inverse document frequency* ($tf \times idf$) measure. After the sort, terms that are more important (i.e., having higher $tf \times idf$ values) are selected to form the *concepts* set.

Now we must convert the vocabulary (set of concepts) into a Bayesian network and define the initial conditional probabilities. To do this, we create a Bayesian network (Hecherman 1995; Mitchell 1995; Witten and Frank 2000), whose nodes are the concepts of the vocabulary. We organize this network in such a way that the more specific concepts are the parents of the more general concepts. This process leads to a set of directed acyclic graphs (DAGs). An example of this organization is shown in Figure 2.

As depicted in Figure 2, we must establish some relations. This is an important concept in Bayesian networks, and is known as *Learning the Bayesian Network* (Hecherman 1995; Witten and Frank 2000). A very popular, simple and fast algorithm for learning the Bayesian networks is *K2* (Witten and Frank 2000). This algorithm imposes a kind of ordering on the nodes and processes the nodes one by one in order, adding edges from previously processed nodes to the current one in a greedy manner. In each step, the edge that maximizes the score of the network is added. When there is no further possible improvement, the next node is selected and the process is continued. Because only the edges of the previously processed nodes are considered and there is a fixed ordering, this procedure cannot lead to any cycle. However, the result of this algorithm depends on the initial ordering, thus it is necessary to run several instances of the algorithm with different orderings. We improve this algorithm by considering the particular characteristics of the Web documents. In contrast to of the greedy behavior of *K2*, we create a relation only if the association between two nodes (concepts)

has a certain degree of confidence. The possibility of *over-fitting* (which is possible in *K2*) is thus prevented. Furthermore, a relation is always created from the more specific concept (with smaller number) to the more general concept (with greater number), thus the network cannot have cycles. Using these enhancements, there is no need for several instances of the algorithm. In particular, for each pair of concepts in the network, the possibility of cooccurrence in different pages is calculated. If the frequency of the cooccurrences satisfies condition (1), it is considered a *good* occurrence. Finally, the relationship (joint probability) of two concepts is calculated by means of (2), and if it exceeds a predefined threshold, a relation is created from the more specific concept as the parent to the more general concept as the child, thus the base relations are created with a predetermined level of confidence.

$$(n_1 \geq 0.3n_2 \text{ or } n_2 \geq 0.3n_1), \quad n_1, n_2 > 0 \tag{1}$$

$$\frac{\#of\ good\ occurences}{\#\ of\ all\ docs} \tag{2}$$

After creating the initial relations, we should complete the conditional probability tables. While in reality we want to maximize the *conditional* probability, in most cases there is no closed-form solution for the maximum conditional-likelihood probability estimates. Hence it has been proposed to use standard maximum joint-likelihood probability estimates in the network (Witten and Frank 2000). A useful trick is to use a *Markov blanket* (Hecherman 1995; Mitchell 1995). The *Markov blanket* method is based on the conditional independence of a node from all other nodes given the values of the nodes in their Markov blanket. The Markov blanket of a node includes all its parents, children, and the parents of children. Here we compute the joint probability $\frac{P(A,B_1,B_2,\ldots,B_n)}{P(B_1,B_2,\ldots B_n)}$ to estimate the value of $P(A|B_1, B_2, \ldots, B_n)$, where the B_i's are probabilistic variables with values of *yes* or *no*. Therefore for each node (concept) A, the joint probability of A with its parents is calculated, and then the joint probability of the parents of A is calculated separately. Dividing the first value by the second one yields the probability $\frac{P(A,B_1,B_2,\ldots,B_n)}{P(B_1,B_2,\ldots B_n)}$. This is continued for different values of the binary probabilistic variables $B_1, B_2, \ldots, B_n$. In general, as is shown in formula (3), we should have to compute two classes of probabilities. Given one probability, the other is the complement of the first. Therefore, after calculating the first probability, we can obtain the second probability by $1 - P(A = yes|B_1, B_2, \ldots, B_n)$. We assume that a Web page contains a concept if the relation (4a)[2] holds. This relation is used in (4b) to calculate the probability $P(A = yes|B_1, B_2, \ldots, B_n)$ (probability 3[1]) and its complement is the probability $P(A = no|B_1, B_2, \ldots, B_n)$ (probability 3[2]).

$$\begin{aligned} &(1) \quad P(A = yes|B_1, B_2, \ldots, B_n) = \frac{P(A = yes, B_1, B_2, \ldots, B_n)}{P(B_1, B_2, \ldots, B_n)} \\ &(2) \quad P(A = no|B_1, B_2, \ldots, B_n) = 1 - P(A = yes|B_1, B_2, \ldots, B_n) \end{aligned} \tag{3}$$

[2] The constant values of (1) and (4) can be easily determined by some limited human-oriented examinations, and approximate values are sufficient.

(a) *Web page contains the concept AND* $W_{tf/idf}(concept, web\ page) \geq 0.3W_{\max}$ $(web\ page)$

$P(A = yes|B_1, B_2, \ldots, B_n)$

$$\text{(b)} = \frac{\dfrac{Number\ of\ pages\ containing\ A\ and\ containing\ (not\ containing)\ B_1\ and\ \ldots}{Number\ of\ all\ pages}}{\dfrac{Number\ of\ pages\ containing\ (not\ containing)\ B_1\ and\ \ldots}{Number\ of\ all\ pages}} \tag{4}$$

$$= \frac{Number\ of\ pages\ containing\ A\ and\ containing\ (not\ containing)\ B_1\ and\ \ldots}{Number\ of\ pages\ containing\ (not\ containing)\ B_1\ and\ \ldots}$$

After constructing the Bayesian network, the nodes (concepts) that are placed at lower levels and contain more general semantics are selected as the members of the *category* set. More precisely, the category set is formed as follows:

(1) Search the Bayesian network.
(2) Select the nodes without children (leaves).
(3) Select the nodes with more direct or indirect parents.
(4) If necessary, limit the selection to a specific number (e.g., 25 nodes).

We propose to select these nodes as *categories* because they represent more general concepts that are more complete representatives for the domain and enable us to reduce the dimensionality of the vector space.

3.2. Grouping the Categories and Finding Cluster Centers

After constructing the Bayesian network and finding the categories, we group the categories. To do this, at first we represent each category by a vector with dimensions equal to the number of concepts in the network. The value of each dimension is the computed relationship of the category with the concept representing the dimension. At the end, we obtain a vector for each category. Then we must still calculate the relationship between each category and each concept.

To do this, for all pairs of the form {"*concept*" (concepts that are not categories), "*category*"} we obtain the direct paths from "*concept*" to "*category*" and eliminate all the nodes that are not in the path. Then we set the prior probability of the root node ("*concept*") to be 1 and by inferring from the Bayesian network, obtain the belief of the last node ("*category*"). This process, depicted in Figure 3, is repeated for all categories. Figure 4 shows an example of this algorithm in which the nodes that are not on the paths are ignored.

There are several methods of grouping the categories. Here, the number of categories is finite and bounded, thus the main challenge deals with the quality of the groups, and the time complexity is not so important. We follow a process that is similar to the average linkage hierarchical method (Day and Edelsbrunner 1985) but produces results in a flat format. The method repeatedly combines smaller groups to generate bigger ones. First each category constructs a single group, and then some groups are composed to construct more general groups. The criterion used for combining the smaller groups is to minimize the average distance between the data inside the two groups. For this purpose, the algorithm, in each iteration, calculates the average distance between any two groups by averaging the sum of all pairwise distances inside two groups, and then combines the two groups with the minimum average distance. Although the time complexity of this algorithm is $O(c^3)$ (c is a constant

Algorithm 2 (Inferring the relationship between each category and the concepts of the network):

1. **for each** *category* **in** *network* **do:**
2. **for each** *concept* **in** *network* **do:**
2.1. In *network*, find all *paths* between *concept* and *category*;
2.2. Eliminate the nodes that do not exist in *paths*;
2.3. Set the belief of the *path*'s root node (*concept*) to 1;
2.4. Infer the belief of the *path*'s last node (*category*);

FIGURE 3. Inferring the relations between the categories and concepts.

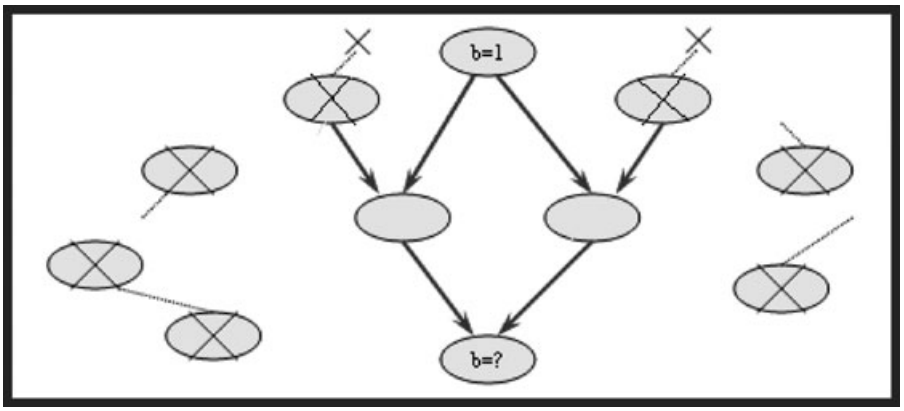

FIGURE 4. An example from extracting the relation between a concept and a category.

value representing the number of categories), the resulting clusters have high quality. Note that these groups will be used as cluster centers, thus their quality levels have an important impact on the quality of the clustering solution.

Now we construct a vector for each cluster center with dimension equals to the number of categories. We set values[3] equal to 20 or 0 for each dimension. The value is set to 20 when the cluster contains the category; otherwise 0 is selected. Therefore, at the end, we extract the representative point of each cluster in a knowledge extraction process and now we should assign the Web pages of the test collection to the appropriate clusters.

3.3. Assigning the Web Pages to the Cluster Centers

After determining the cluster centers, the only remaining task is to assign each Web page of the test collection to a suitable center. To calculate the similarity between each page and the cluster centers, we must construct a vector for each Web page with dimension equal to the number of categories. Each coordinate of this vector represents the relationship between the Web page and the associated category. To construct this vector, for each Web page, after elimination of stopwords and stemming, a vector with dimension equal to the number of concepts is created whose coordinates represent the weights of the associated concepts in the Web page. The terms that do not exist in the network are ignored.

[3] The value 20 is selected based on our experiments. For other domains, an expert should select an appropriate value.

Then, each vector is reduced to another vector with dimension equal to the number of categories. For each vector, the algorithm proceeds as follows: for each concept, the belief of each category with having complete belief to the concept is inferred. To do this, as shown in Figure 3, the effect of the irrelevant nodes on the inference is ignored and the result vector is updated by "incremental adding" for all concepts. This process is repeated for all pages.

For assigning Web pages to clusters, we use cosine similarity with a predefined threshold; if the similarity of a Web page with a cluster center is more than this threshold, the Web page will be assigned to the cluster.

4. IMPROVING TOWARD HIERARCHY

So far, we have proposed a method for extracting cluster centers at the same level. Now, we extend the method to hierarchical clustering. To do this, our improvement includes two stages:

(1) Constructing the cluster centers in a hierarchical manner.
(2) Assigning each Web page of the test data to a suitable cluster at an appropriate level.

We now describe each step in detail. The method of grouping categories is similar to what was proposed in Section 3 but creates clusters in a hierarchical format. To do this, we first define a threshold; if the similarity between the two nearest clusters exceeds this threshold, we join them as a single cluster; otherwise they are joined at a higher level. Because these groups become larger and larger, considering a constant threshold does not seem suitable. Earlier combinations require larger threshold values than later combinations. Therefore, we define a measure according to (5) and try to optimize it during the combination process.

$$\frac{Sum\ of\ Distances\ of\ G_i + Sum\ of\ Distances\ of\ G_j + The\ Distance\ Between\ Two\ Groups}{(n_i + n_j - 1)(The\ Distance\ Between\ Two\ Groups)} ?\ T \quad (5)$$

Here T is the grouping threshold and n_i is the size of group G_i (notice that the number of distances in the compound group is equal to $n_i + n_j - 1$). This measure has been defined according to this assumption that the new distance can affect the density of the groups, thus its influence should be examined, while the previous distances have already been examined. According to the result of relation (5) we are faced with two situations:

(1) If $LeftSide \geq T$, then the two groups join to each other and construct a single group at the same level. The new distance is added to the set of distances of the new group, and the other parameters of the new group (such as number of members, etc.) are updated.
(2) If $LeftSide < T$, then a new group is constructed at a higher level and these two groups are set as its children. This new group has two members and the new distance is the set of its distances.

Here, a main concern is the value of this threshold, which can affect the quality of the grouping. We therefore describe a heuristic method for selecting the best of several candidate hierarchies using a predefined measure. The measure is called "*balancing the hierarchy*" and is defined in Definition 3.

Definition 1. The number of unbalanced documents in a node (cluster) is defined as the absolute value of the difference between the number of documents of the left child and the number of documents of the right child.

Definition 2. The number of unbalanced documents in a hierarchy A is defined as the sum of number of unbalanced documents in all of its nodes.

Definition 3. We say that hierarchy A is more balanced than hierarchy B if for a predefined number of documents, the number of unbalanced documents in A is less than (or at most equal to) the number of unbalanced documents in B. In other words,

$$\frac{Number\ of\ Unbalanced\ Documents\ \text{in}\ A}{Number\ of\ Unbalanced\ Documents\ \text{in}\ B} \leq 1.$$

Of course, some exceptional situations may occur when the denominator is equal to 0. In this situation, if the numerator is 0, hierarchy A is considered to be the more balanced hierarchy, and otherwise hierarchy B is considered to be the more balanced hierarchy.

Definition 4. Node a is k-unbalanced if the absolute value of the difference between the number of documents in its left child and the number of documents in its right child is equal to k.

We prove that if we want to have a hierarchy for which the probability of wrong assignments of documents (test data sets) is minimized, we should construct the hierarchy in accordance with the *No free lunch*[4] theorem (Wolpert and Macready 1997). Then we prove that considering this theorem results in the selection of a more balanced hierarchy. For this purpose, we need to define a relationship between the *No free lunch* theorem and the balancing of the hierarchies.

Lemma 1. Assume that node a is k-unbalanced for $k = i$ and has two children. The next value of k for which a is k-unbalanced is $k = i + 2$.

Proof. Assume that n data points are forwarded from node a to its children. Between these data points, k points are dedicated to only one child and the remaining $n - k$ data points are divided between the two children. Then $n - k$ must be an even number to be divisible between the children. If n is an even number, then k must also be an even number, and if n is odd, then k must also be an odd number. For both of these situations, for a given value of n, the difference between any two consecutive values in a sequence of k-unbalanced values is 2. ■

Theorem 1. Assume that we are given a set of hierarchies and that identical data points are assigned to all of them. The probability that a more balanced hierarchy is consistent with the *No free lunch* theorem is higher than the probability of the consistency of a less balanced hierarchy. (In other words, the probability that the *No free lunch* theorem leads to a more balanced hierarchy is higher than the probability of its yielding a less balanced hierarchy).

Proof. Assume that hierarchy h has already been constructed. For convenience, without loss of generality, assume that each node has either two or no children (the proof is simply extendable for hierarchies with other numbers of children). During the assigning phase, each node is examined to determine if the incoming data object must be assigned to it or must be forwarded to the child nodes. Assuming that we are currently in node a, there are two possible situations:

[4] The *No free lunch* theorem states that in the absence of specific assumptions on the nature of data, then on average there is no difference between various classifiers applied to the data. This theorem is proved by averaging over all data states in a specific classifier.

(1) The data point is assigned to node a, thus no change occurs in balancing the hierarchy.
(2) The data point is forwarded to one of the right (r) or left (l) children.

In the second situation, we show that according to the *No free lunch* theorem (Wolpert and Macready 1997) the probabilistic expected value of the number of data points moved toward r is equal to the corresponding value for l. According to the *No free lunch* theorem, there is no optimal classifier, and different classifiers have no precedence over others. Therefore, the number of classifiers that move the data point to r is equal to the those that move the data point to l. Thus the probability of the movement of the data point to r is equal to $\frac{\textit{Number of classifiers move the data to right}}{\textit{Number of all classifiers}} = \frac{1}{2}$. A similar relation holds for l. Thus the expected number of data points moved to each child is $\sum_{i=1}^{E(n)} \frac{1}{2} = \frac{E(n)}{2}$, where $E(n)$ is the expected value of the number of data points forwarded from node a (this shows that *No free lunch* theorem applies a kind of balancing to the clustered structure).

In what follows, we calculate the probability of k-unbalancing for different values of k in node a. We assume that n data points are forwarded from node a (node a has two children). In this way, $\frac{n+k}{2}$ data points are moved to one node and the remaining $\frac{n-k}{2}$ are placed in the other node. Because the probability of assigning a data point to a child node is $\frac{1}{2}$, the probability of a k-unbalancing in node a can be computed according to (6):

$$\begin{aligned} P(x = k) &= \binom{n}{\frac{n+k}{2}} \left(\frac{1}{2}\right)^{\frac{n+k}{2}} \left(\frac{1}{2}\right)^{\frac{n-k}{2}} \\ &= \frac{n!}{\left(\frac{n+k}{2}\right)!\ \left(\frac{n-k}{2}\right)!} \left(\frac{1}{2}\right)^{n}. \end{aligned} \tag{6}$$

Now we compare the probability of (6) for different values of k. According to Lemma 1 (that the difference of two consecutive k-unbalanced values is 2), we get (7):

$$\begin{aligned} \forall\, 0 \le i < i+2 \le n:\ \frac{P(x=i)}{P(x=i+2)} &= \frac{\dfrac{n!}{\left(\frac{n+i}{2}\right)!\ \left(\frac{n+i}{2}\right)!}\left(\frac{1}{2}\right)^{n}}{\dfrac{n!}{\left(\frac{n+i+2}{2}\right)!\ \left(\frac{n-i-2}{2}\right)!}\left(\frac{1}{2}\right)^{n}} \\ &= \frac{\left(\frac{n+i}{2}+1\right)!\ \left(\frac{n-i}{2}-1\right)!}{\left(\frac{n+i}{2}\right)!\ \left(\frac{n-i}{2}\right)!} \\ &= \frac{\left(\frac{n+i}{2}+1\right)}{\left(\frac{n-i}{2}\right)} > 1 \end{aligned} \tag{7}$$

We therefore conclude (8):

$$\forall 0 \leq i < i+2 \leq n: \quad P(x=i) > P(x=i+2) \tag{8}$$

These conclusions can be proved for other nodes in a similar way. Therefore the *No free lunch* theorem results in the selection of a more balanced clustering structure.

Theorem 2. Consider hierarchies $h_1, h_2, \ldots$, and assume that identical data sets (*training* data sets) are assigned to all of them. We know that each hierarchy was constructed from some correct and real patterns (patterns that are applicable to other data sets) and some incorrect (unreal) patterns. We do not know which patterns are correct, and even their sizes are unknown. In this situation, to decrease the probability of an inefficient clustering in other data sets (*test* data sets), the more balanced hierarchies should be selected with higher probabilities.

Proof. In this proof we assume that each hierarchy is constructed from correct and incorrect patterns, and we try to investigate the probabilistic effect of each part on balancing the hierarchy.

We have no information about the positions or the sizes of the correct patterns in each hierarchy. Thus we cannot precisely determine the effect of the correct patterns on the balancing or unbalancing of the hierarchy. Moreover, in general, we cannot identify which of a set of separate hierarchies $h_1, h_2, \ldots$ has more correct patterns. Therefore, we can conclude that on average, the probability of affecting the correct patterns on balancing a hierarchy will be identical for the other hierarchies. This is because there are no reason to prefer the correct patterns of one hierarchy to the other one's (in general we cannot even claim that the correct patterns of hierarchy h increase or decrease the balancing of the hierarchy, thus the probability of the hierarchy sloping to the right is equal to that of it sloping to the left).

Therefore, what remains is to examine the effect of the parts of hierarchies that are not included in the correct patterns on the precision of the test data sets. Because these parts are not real patterns, they should not cause any affect in sloping the data to some specific patterns (paths of the hierarchy). This means that there is no specific classifier in these parts to affect the data, because if there existed such a classifier (pattern), this classifier would be unreal (incorrect) and only create *overfitting*, which decreases the precision of the clustering. This conclusion is a confirmation of the *No free lunch* theorem.

In following, we must show that establishing the *No free lunch* theorem follows this fact that the probability of creation of a more balanced hierarchy is greater than the probability of creating a less balanced one. This was proved in Theorem 1. We can thus conclude that to choose the one out of several hierarchies that maximizes the probability of creating appropriate clusters (clusters with higher probability of being accurate), we should select a more balanced hierarchy with a higher probability. ■

So far, in using these theorems, we can choose a hierarchy (the more balanced hierarchy) and can hope that it will be the desired hierarchy. In the next stage, we must assign each Web page of the test collection to the cluster centers. To do this, we define a measure and try to optimize it during the assignment process. The measure should not be sensitive to the number of members of the groups, thus the cosine similarity is selected. The assigning process is shown in Figure 5.

Each Web page traverses a path with length at most equal to the depth of the hierarchy, and each level is compared with the level's (current) node and its children (to calculate the similarity). Thus the complexity is $O(dc)$, where d is the depth of the hierarchy and c is the average number of children of a hierarchy node. Therefore the total complexity (with size N) is $O(dcN)$, and if we ignore the constants d and c it is $O(N)$.

Algorithm 3 (Assigning web pages to suitable cluster centers):

```
1.    for each web page in dataset do:
1.1.      Set currentNode to root;
1.2.      while (true)
1.2.1.       Find the similarity between currentNode and web page;
1.2.2.       Set maxSim by root and similarity;
1.2.3.       for each child in currentNode.childs do:
1.2.3.1.        Find the similarity between child and web page;
1.2.3.2.        if it is greater than maxSim.Sim then:
1.2.3.2.1.         Update maxSim by child and new similarity;

1.2.4.       if currentNode == maxSim.Node then:
1.2.4.1.        Exit from while loop;
1.2.5.       else:
1.2.5.1.        Set currentNode to maxSim.Node;

1.3.      if maxSim.Sim > T then:
1.3.1.       Assign web page to maxSim.Node;
```

FIGURE 5. Finding the most relevant cluster center for each Web page.

5. EXAMINATION AND EVALUATION OF THE PROPOSED METHOD

For implementation settings, first a domain is selected. We have selected the *Politics* domain and restricted the clustering to the Web pages that are related to political news and documents. We collected different Web pages from this area. For this purpose, we performed some queries starting with some general terms and referred to well-known News agencies such as CNN and BBC to collect initial pages. From observations of the collected pages, we could get more accurate knowledge about the domain and make our queries more accurate. We repeated the process again and again until we obtained a suitable set of Web pages related to the area as well sufficient expertise about the more important terms and concepts belonging to this area. Finally we made a (and for some data sets more than one) clustering of the suitably collected Web pages as a reference for our experiments.

5.1. Examination of the Proposed Method of Flat Clustering

To construct a vocabulary that contains the *Politics* area, the following steps were implemented.

(1) A set of Web pages was collected as the training set (500 documents).
(2) The terms were extracted and ordered in descending order according to their *tf-idf* values, and the first 124 of them[5] were selected as the set of concepts.
(3) Each concept was converted into a node in the Bayesian network and the nodes were organized from more specific concepts to more general concepts.
(4) While creating the initial relations, we selected the constant value 0.65 for relation (2).[6] Then the conditional probability tables were completed.

[5] A considerable difference of *tf-idf* values was observed between the first 124 terms and the remaining terms.

[6] The constant value can be easily determined by some limited human examination.

TABLE 1. Categories Obtained from the Bayesian Network.

ADMINISTRATION	EAST	MUSLIM	ELECTION	THREAT
GOVERNMENT	OIL	POLITIC	PRESIDENT	PEOPLE
REVOLUTION	WAR	LEADER	RELIGIOUS	PARTY
DEMOCRACY	WEST	TERROR	ECONOMIC	COURT
PARLIAMENT	WIN	ENERGY	COUNTRY	WORLD

After completing the network, concepts are ordered, and a total of 25 categories[7] were selected, which are shown in Table 1.

Now, we group the categories using the aforementioned grouping algorithm. The results of grouping the categories for different repetitions (number of combinations) are depicted in Figure 6. It is obvious that as the number of repetitions increases, more base groups are combined and more general cluster centers are created. Thus we can control the granularity of the clustering by choosing different repetition numbers. Moreover, because the categories do not represent the concepts equally, it is reasonable that the combinations do not occur identically.

Then we selected 220 test Web pages randomly from the *Politics* domain. For each Web page, a vector with 25 dimensions (the number of the categories) was created and assigned to the nearest cluster center. The cosine similarity between each vector and each cluster center was calculated. If the result was more than 0.75,[8] the Web page was added to the cluster. For a more precise examination, we compare the situation in which the algorithm was run for 10 iterations and that in which the algorithm was run for 14 iterations. The sizes of the clusters for both situations are presented in Figure 7.

The more rapidly a cluster center grows (the more groups are joined together to construct it), the more pages it absorbs during the assignment phase. For example, tracing the process of Figure 6 shows that cluster "*COUNTRY* + *GOVERNMENT* + *POLITIC*" has high ability for combination and thus grows extremely rapidly. As seen in Figure 7, this cluster includes many Web pages.

Another point is the method of constructing the cluster center vectors and the reason for selecting the value of 20. It might have been thought that the selected value could affect the results of the clustering. For a more thorough examination, we implemented the clustering with values 10, 15, 20, 25, and 30; the experimental results were exactly the same for all values. This shows that the selected value does not affect the clustering, and the method is not sensitive to this parameter.

5.2. Evaluation of Clustering Results

Several methods have been proposed to evaluate clustering results. The most famous methods are the entropy-based Quinlan (1993) and F-measure (Larsen and Aone 1999). In the first method, the entropy of a cluster is calculated by $E(j) = -\sum_i pij \log(p_{ij})$, where i denotes the correct class and j is the cluster produced. Then the total entropy is $E_{CS} = \sum_{j=1}^{m} \frac{n_j \times E(j)}{n}$. A good clustering algorithm minimizes the total entropy.

[7] Again, this threshold is determined by an expert.

[8] The value of this threshold is a subjective issue that depends on the human perceptions of relevancy, precision, and recall. It can be easily determined by some limited human-oriented examinations.

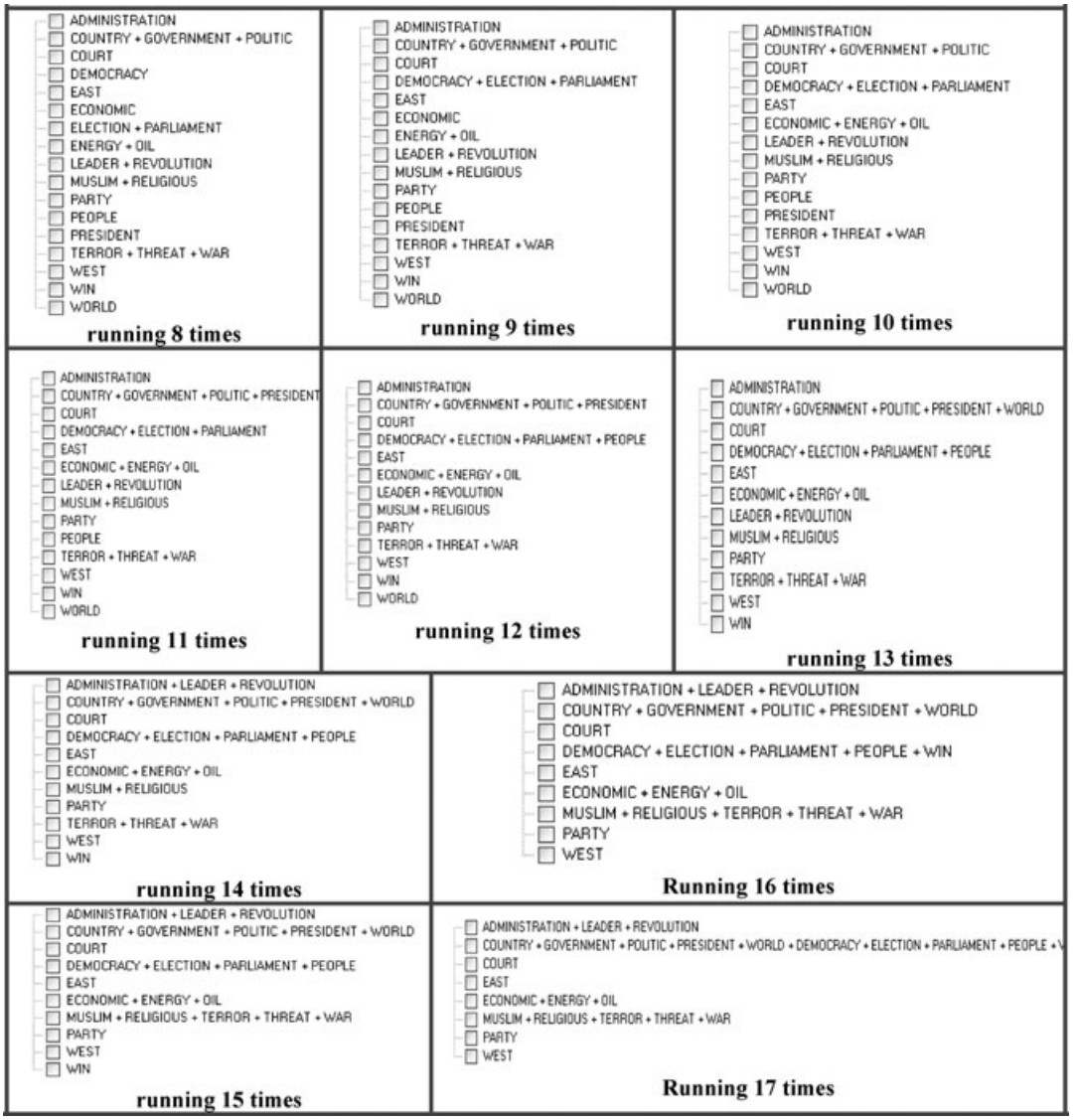

FIGURE 6. The cluster centers resulting from different numbers of iterations.

The F-measure combines two measures, namely *precision* and *recall*, and evaluates whether the clustering can remove noise pages and generate clusters with high quality. If P and R denote the *Precision* and *Recall*, respectively, this measure is defined by (9), where the precision and recall are obtained by (10). In these formulas n_j denotes the size of cluster j, g_i the size of class i, and N (i,j) the number of pages of class i in cluster j.

$$F(i,j) = \frac{2(P(i,j) * R(i,j))}{P(i,j) + R(i,j)}, F = \sum_i \frac{g_t}{n} m_j ax\{F(i,j)\} \tag{9}$$

$$P(i,j) = N(i,j)/n_j, R(i,j) = N(i,j)/g_i \tag{10}$$

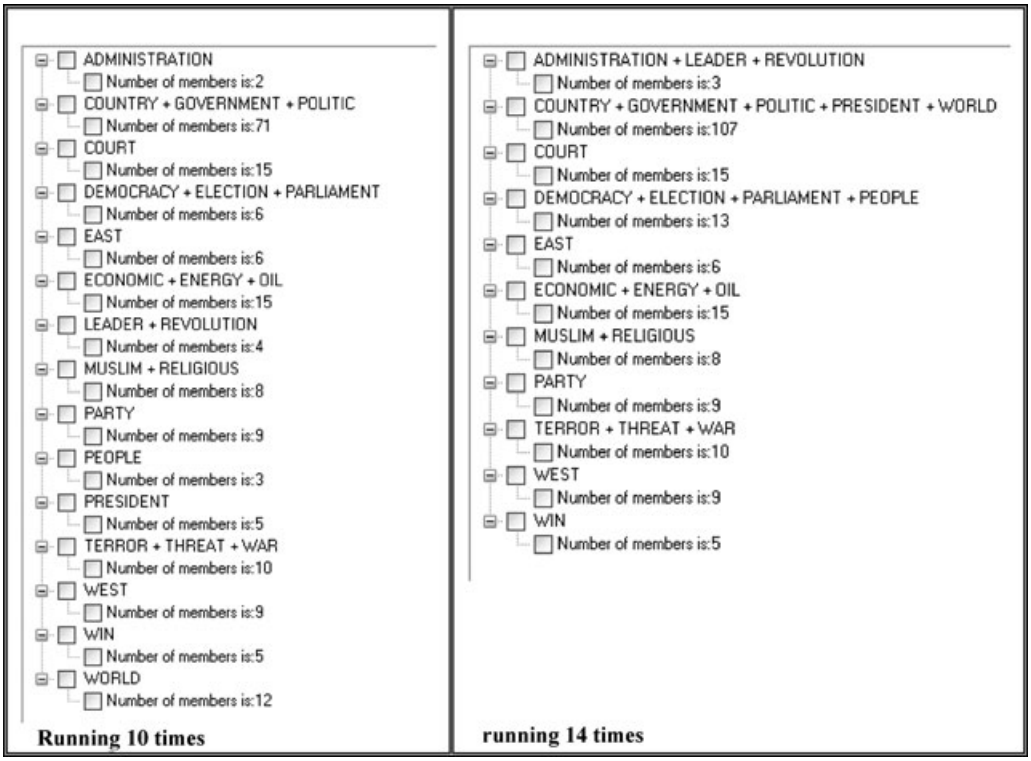

FIGURE 7. Comparing the numbers of cluster members when running 10 and 14 iterations.

Because of the popularity of the F-measure for the evaluation of Web page clustering, we use it for our evaluations. For the proposed method, in most cases the F-measure of each cluster has a good value. For example, the value of the total F-measure when running the algorithm for 10 iterations is 0.84868 and its value on 14 iterations is 0.86534. These values and their associated precision and recall values are good signs for high quality of the proposed clustering process. For a more precise comparison, we have implemented the K-Means and have applied that method to the same data set. The results are presented in Table 4. Comparing the results of Tables 2 and 3 with Table 4 shows the efficiency of the proposed method. The total F-measure value of the K-Means is 0.65707, which indicates a considerable improvement in the proposed clustering process.

In Tables 2 and 3, the values of the precision are promising. The main reason for the somewhat lower values of the recall is that we have set the threshold of cosine similarity to a relatively high value (0.75). If we decrease the threshold, the recall can be increased (on the other hand, we may have some reduction in precision). Instead, with the selected threshold, the pages that are assigned to the clusters have more relatedness and similarity to the cluster topic, and in fact a kind of priority is applied to the more related pages.

In what follows, we examine the method with more test collections. The results are summarized in Table 5. Each data set is collected randomly and independent of the other data sets from the *Politics* domain. By comparing the F-Measure values of Table 5, we see clearly that the proposed method produces clusters with higher quality.

TABLE 2. Evaluation of Clustering when Running the Algorithm for 10 Iterations.

Cluster	N	P	R	F
ADMINISTRATION	2	1	0.38	0.550724638
COUNTRY+GOVERNMENT+POLITIC	71	0.9	0.92	0.90989011
COURT	15	0.92	0.82	0.867126437
DEMOCRACY+ELECTION+PARLIAMENT	6	0.83	0.6	0.696503497
EAST	6	0.83	0.63	0.71630137
ECONOMIC+ENERGY+OIL	15	0.91	0.79	0.845764706
LEADER+REVOLUTION	4	0.94	0.58	0.717368421
MUSLIM+RELIGIOUS	8	0.88	0.67	0.760774194
PARTY	9	0.82	0.81	0.814969325
PEOPLE	3	1	0.48	0.648648649
PRESIDENT	5	0.94	0.72	0.815421687
TERROR+THREAT+WAR	10	0.85	0.74	0.791194969
WEST	9	0.86	0.87	0.864971098
WIN	5	0.9	0.67	0.768152866
WORLD	12	0.91	0.93	0.919891304

TABLE 3. Evaluation of Clustering when Running the Algorithm for 14 Iterations.

Cluster	N	P	R	F
ADMINISTRATION+LEADER+REVOLUTION	3	1	0.54	0.7012987
COUNTRY+GOVERNMENT+POLITIC+PRESIDENT+WORLD	107	0.84	1	0.9130434
COURT	15	0.92	0.82	0.8671264
DEMOCRACY+ELECTION+PARLIAMENT+PEOPLE	13	0.87	0.73	0.793875
EAST	6	0.83	0.63	0.7163013
ECONOMIC+ENERGY+OIL	15	0.91	0.79	0.8457647
MUSLIM+RELIGIOUS	8	0.88	0.67	0.7607741
PARTY	9	0.82	0.81	0.8149693
TERROR+THREAT+WAR	10	0.85	0.74	0.7911949
WEST	9	0.86	0.87	0.8649710
WIN	5	0.9	0.67	0.7681528

TABLE 4. Evaluation of *K*-Means with 10 Centers.

Center ID	N	P	R	F
1	14	0.65	0.59	0.61854838
2	4	0.49	0.53	0.50921568
3	21	0.71	0.67	0.68942029
4	26	0.57	0.71	0.63234375
5	15	0.48	0.56	0.51692307
6	3	0.73	0.71	0.71986111
7	76	0.68	0.72	0.69942857
8	17	0.63	0.71	0.66761194
9	31	0.59	0.67	0.62746031
10	13	0.66	0.74	0.69771428

TABLE 5. The Results of Applying the Method on Different Test Sets.

Data set	Data set size	Number of clusters	Proposed method	K-Means
Data set 2	300	10	0.83542	0.67352
Data set 3	400	12	0.85362	0.62158
Data set 4	500	14	0.81453	0.63747
Data set 5	600	14	0.87912	0.71650
Data set 6	700	15	0.82141	0.68630

TABLE 6. Comparison of Different Hierarchies with Different Values for T.

T_A	0.5	0.55	0.6	0.65	0.7	0.75
T_B	0.55	0.6	0.65	0.7	0.75	0.8
A/B	0.896	0.932	0.961	1.171	99.023	1.158

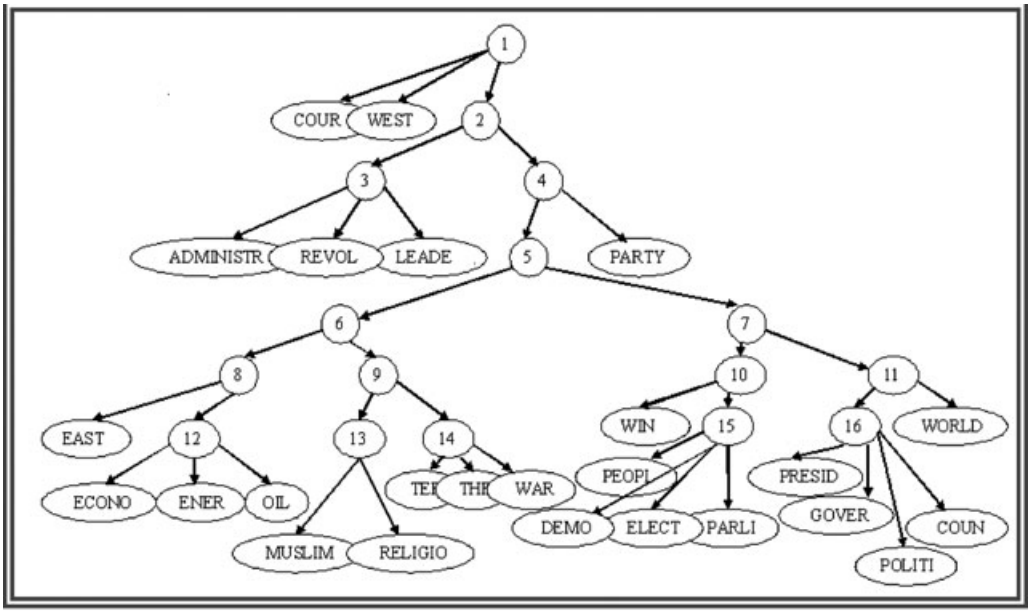

FIGURE 8. A hierarchical schema from cluster centers.

5.3. Examination of Hierarchical Clustering

In this section we examine the hierarchical method and evaluate its efficiency. First we construct the hierarchy of cluster centers. For this purpose, we define the threshold T with values 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, and 0.8 to construct the hierarchy (in combinations of groups) and select the best result as the final hierarchy. The results are shown in Table 6.

The best result is obtained with $T = 0.65$ (this value yields a better hierarchy even than $T = 0.75$). This best hierarchy is depicted in Figure 8 (because of space limitations, some of the category names are truncated). In Figure 8, each intermediate node is a cluster center constructed from the union of the categories of the subcenters. The leaf nodes do not

TABLE 7. Results of Assigning Web Pages to Hierarchical Cluster Centers.

Center ID	Center labels	N	P	R	F
1	WEST+COURT+CLUSTER2	293	0.75	0.84	0.79245283
2	CLUSTER3+ CLUSTER4	274	0.75	0.82	0.78343949
3	ADMINISTRATION+ REVOLUTION+LEADER	18	0.82	0.83	0.824969697
4	PARTY + CLUSTER5	251	0.76	0.81	0.784203822
5	CLUSTER5+CLUSTER7	232	0.77	0.82	0.794213836
6	CLUSTER8+CLUSTER9	107	0.88	0.86	0.869885057
7	CLUSTER10+CLUSTER11	115	0.75	0.79	0.769480519
8	EAST+CLUSTER12	39	0.91	0.90	0.904972376
9	CLUSTER13+CLUSTER14	54	0.86	0.80	0.828915663
10	WIN+CLUSTER15	46	0.76	0.79	0.774709677
11	WORLD+CLUSTER16	57	0.83	0.81	0.819878049
12	ECONOMIC + ENERGY + OIL	25	0.93	0.91	0.919891304
13	MUSLIM + RELIGIOUS	19	0.88	0.83	0.854269006
14	TERROR+THREAT+WAR	26	0.86	0.76	0.80691358
15	PEOPLE+DEMOCRACY+ ELECTION+PARLIAMENT	33	0.79	0.80	0.794968553
16	PRESIDENT+GOVERNMENT+ POLITIC+COUNTRY	45	0.85	0.81	0.829518072

construct cluster centers alone. The final clusters are depicted in Table 7 (first and second columns).

After constructing the hierarchy, we choose another data set (test data) collected randomly from *Politics* sites. After creating their vectors, we assign each Web page to the appropriate cluster center according to the algorithm of Figure 5.

Evaluation methods such as F-measure and entropy have been developed for evaluation of flat clusters (or the lowest levels of hierarchical clusters). Thus we must adapt the F-measure for evaluation of hierarchical clustering algorithms. To do so, we first apply the F-measure on the clusters without any children (at the lowest levels) and then compute the F-measure of the higher clusters according to the values of their children. Formula (11) defines the way of computing the *Precision* of a parent cluster recursively based on the *Precision* of its children.

$$P_c = \sum_{\forall k \in C.\,children} \frac{n_k}{n} P_k + \frac{n - \sum_{\forall k \in C.\,children} n_k}{n} P_R \tag{11}$$

$$R_c = \sum_{\forall k \in C.\,children} \frac{n_k}{n} R_k + \frac{n - \sum_{\forall k \in C.\,children} n_k}{n} R_R \tag{12}$$

In formula (11), P_R shows the *Precision* of the cluster members that do not belong to any other cluster. The *Recall* can be computed in a similar way, as shown in formula (12). However, because for computing the recall we must calculate the R_R by traversing all

TABLE 8. Results of Hierarchical Clustering for Different Algorithms.

Data set	Data set size	Proposed method	Single linkage	Average linkage
Data set 1	300	0.80205	0.61812	0.60952
Data set 2	500	0.81728	0.58943	0.61507
Data set 3	1000	0.77381	0.52701	0.57195
Data set 4	2000	0.78115	0.53719	0.55389

(relevant) Web pages, it is simpler to calculate the recall separately and without considering precalculated values for subclusters. Therefore we follow a button-up process to find the F-measure values: first we calculate the F-measure, *Precision* and *Recall* values of the clusters with no children and then compute the F-measure of each parent cluster from the parameters of its children. Using this process, the results are shown in Table 7. The total F-measure for this case is 0.80205, which seems acceptable for a hierarchical clustering.

We now examine hierarchical method with more test collections. The results are summarized in Table 8. Each data set was collected randomly and independently of the other data sets. The results of Table 8 are other demonstrations of the effectiveness of the proposed method. To provide a more detailed view of the efficiency of the proposed method, we have compared it with two well-known agglomerative hierarchical methods. Single and average linkage methods are very popular Web page clustering methods that have some similarities to the proposed methods, thus we considered them in our experiments. These methods are examined and evaluated in detail in Zhao and Karypis (2002, 2005).

5.4. Further Experiments

We now considered other methods for our experiments. We have selected the methods proposed in Wang and Kitsuregawa (2002), Zamir and Etzioni (1998), and Steinbach et al. (2003) and compared the clustering results. Wang and Kitsuregawa (2002) extend the standard *K-Means* based on combination of link and content information. The method which is called LCComb here, works as follows (Wang and Kitsuregawa 2002):

(1) Assign each relevant page to the Top *C existing cluster(s)* based on the similarities (above a threshold) between the page and the correspondent centers;
(2) The page is itself one cluster if no *existing cluster* meets step 1;
(3) Recompute the centers of the clusters if their members have changed;
(4) Repeat Steps 1 to 3 until all relevant pages are assigned and all centers are stable;
(5) Merge two base clusters produced by step 4 if they share more members than the merge threshold.

As in the original research, we select the similarity threshold 0.1 and the merge threshold of 0.75 in our experiments. The results are shown in Table 9.

The other method we examine is a phrase-analysis algorithm called Suffix Tree Clustering (STC; Zamir and Etzioni 1998). In essence, the algorithm builds a suffix tree of phrases in documents; each representative phrase becomes a candidate cluster; candidates with large overlaps are merged together. Each base cluster is assigned a score, which is a function of both the number of documents it includes and the number of words that make up its phrases. In fact, the base clusters are clustered using the equivalent of a single linkage clustering

TABLE 9. Average F-Measure Values of Different Methods.

Data set	Data set size	Proposed method	LCComb	STC	SNNC
Data set 1	300	0.80205	0.69832	0.76702	0.73938
Data set 2	500	0.81728	0.73718	0.72155	0.76051
Data set 3	1000	0.77381	0.67425	0.71343	0.74925
Data set 4	2000	0.78115	0.66563	0.68754	0.72093

algorithm where a predetermined minimal similarity between base clusters serves as the halting criterion. The results of applying this method are also shown in Table 9.

Finally, we consider the method proposed in Steinbach et al. (2003). This method begins by calculating the document similarity matrix based on the cosine similarity. Then the nearest neighbor graph is constructed in such a way that there is a link from object i to object j if i and j both have each other in their nearest neighbor list. The details of the shared nearest neighbor clustering algorithm (SNNC) are as follows (Steinbach et al. 2003).

(1) For every point i in the data set, calculate the connectivity, conn[i], the number of strong links the point has (strong links are created according to *strong link threshold*).
(2) For a point i in the data set, if conn[i] $<$ *noise threshold*, then that point is not considered in the clustering because it is similar to only a few of its neighbors. Similarly, if conn[i] $>$ *topic threshold*, then that point is similar to most of its neighbors and is chosen to represent its neighborhood.
(3) For any pair of points (i, j) in the data set, if i and j share significant numbers of their neighbors, i.e., the strength of the link between i and j is greater than the *merge threshold*, then they will appear together in the final clustering if either of them (or both) is chosen to be a representative. Two objects that are not directly related will be put in the same cluster only if there are many other objects between them that are connected with strong links, half of which must represent their own neighborhoods.
(4) Labeling step: Having defined the representative points and the points strongly related to them, we can bring back some of the points that did not survive the *merge threshold*. This is done by scanning the shared nearest neighbor list of all the points that are part of a cluster, and checking whether those points have links to points that do not belong to any cluster and have a link strength greater than the *labeling threshold*.

It is apparent that this method requires several parameters to be set, including the *strong link threshold*, *noise threshold*, *topic threshold*, *merge threshold*, and *labeling threshold*. However determining the optimal values for these parameters is not easy, and in contrast to our method there is no knowledge extraction (training) step to estimate them. To get better results we have determined their values by trial and test. We selected the following values in our experiments: *strong link threshold*: 0.4, 0.5; *noise threshold*: 3,4; *topic threshold*: 7,8; *merge threshold*: 4,5; and *labeling threshold*: 3. The different values of the parameters were chosen based on the size of the data set.

According to the results, the proposed method creates clusters of higher quality than the other examined methods. We have selected these methods because they are advanced methods that try to improve the *K-Means* and link linkage, but as the results show, our improvements are more efficient.

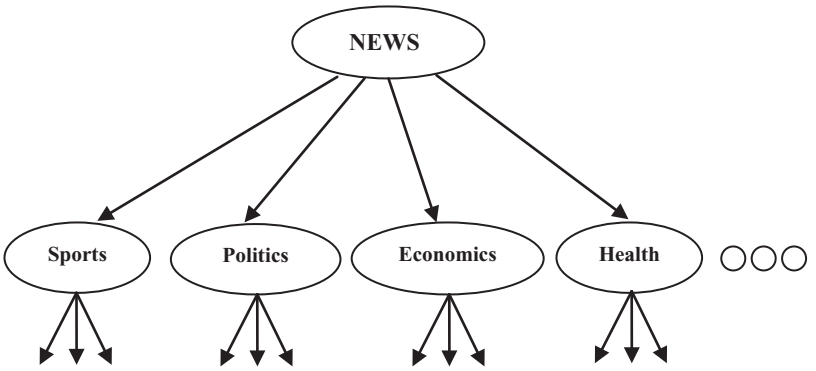

FIGURE 9. The result structure from the preclassification task.

5.5. Scalability Issues

A major question that may arise is the scalability of the proposed method. As mentioned before, the proposed method extracts the knowledge of a preselected domain, thus we must look for the ways to extend the clustering to other domains.

Here we propose two approaches to address this problem. The first approach incorporates a predefined classification into the clustering process. Usually all agree on simple classifications. Therefore, we can consider a simple primary classification and then complete the remaining complex parts by the proposed clustering method. As an example, consider Figure 9. In this figure, a basic structure has been constructed manually, while the remaining parts can be completed by the clustering using the knowledge extracted for each domain (subtree).

The other approach is based on finding the intersections or the similarities between various subspaces. As mentioned before, for each domain, a set of concepts (dimensions of the vector space) is found. Then, after finding all the subspaces, an agglomerative clustering method can be applied to the subspaces to complete the hierarchy.

6. CONCLUSION

In this paper a new method was introduced for clustering Web pages. Like many other Web clustering methods, the proposed algorithm at first selects K representative or cluster center points and then assigns the Web pages to them. The main advantages of the method are (1) it does not need to determine the number of clusters in advance and is less sensitive to parameter settings, and (2) the representative points are not selected at random but instead are computed by a knowledge extraction process.

The main idea of the proposed method is that it constructs a Bayesian network for a preselected domain and provides the ability to draw inferences from the network. Then the method infers the relationships between the concepts and the categories of the domain. To learn the Bayesian network, a method was proposed that identifies the initial relations with confidence and avoids overfitting. In the next step, the categories are grouped using a high-quality algorithm, without considering factors such as scalability and so on. This process also provides a good opportunity for identifying the topics of the clusters.

Then we revised the process to create hierarchical clusters. In the hierarchy construction phase, we applied a theorem on how to select the hierarchy with higher probability of

correctness. This theorem enables us to select better hierarchies with higher probabilities. Finally, the F-measure was changed and adapted for evaluation of hierarchical clusters. In the new definition of the F-measure, the precision levels of the parent nodes are calculated from their children. The experimental results of both the flat and hierarchical methods show improvements in the quality of the result clusters compared to various similar methods.

ACKNOWLEDGMENTS

We would like to express our appreciation to Prof. Caro Lucas for his help during the preparations and the revisions of the paper. It was with deep sorrow that he passed away from cancer on July 8, 2010.

REFERENCES

ABIDI, S. S. R. and J. ONG. 2000. A data mining strategy for inductive data clustering: A synergy between self-organizing neural networks and K-Means clustering techniques. *In* TENCON 2000 Proceedings, Vol. **2**, Kuala Lumpur, Malaysia, pp. 568–573.

BALL, G. and D. HALL. 1967. A clustering technique for summarizing multivariate data. Behavioral Science, **12**:153–155.

BANDYOPADHYAY, S., C. GIANNELLA, U. MAULIK, H. KARGUPTA, K. LIU, and S. DATTA. 2006. Clustering distributed data streams in peer-to-peer environments. Journal of Information Sciences, Elsevier, **176** (14): 1952–1985.

BARTHELEMY, J.-P. and B. LECLERC. 1995. The median procedure for partition, in partitioning data sets, AMS DIMACS Series in Discrete Mathematics, **19**:3–34.

BLOEHDORN, S., and A. HOTHO. 2004. Text classification by boosting weak learners based on terms and concepts. *In* Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM). IEEE Computer Society Press: New York, pp. 331–334.

BOZSAK, E., M. EHRIG, S. HANDSCHUH, A. HOTHO, A. MAEDCHE, B. MOTIK, D. OBERLE, C. SCHMITZ, S. STAAB, L. STOJANOVIC, N. STOJANOVIC, R. STUDER, G. STUMME, Y. SURE, J. TANE, R. VOLZ, and V. ZACHARIAS. 2002. Kaon: Towards a large scale semanticWeb. *In* E-Commerce and Web Technologies, Third International Conference Proceedings, EC-Web 2002, Vol. 2455 of LNCS. *Edited by* K. Bauknecht, A. Min Tjoa, and G. Quirchmayr. Springer: Berlin, pp. 304–313.

BRADLEY, P. and U. FAYYAD. 1998. Refining initial points for K-means clustering. *In* Proceedings of 15th International Conference on Machine Learning (ICML), Madison, WI, pp. 91–99.

CHINRUNGRUENG, C. and SÉQUIN, C., 1995. Optimal adaptive K-means algorithm with dynamic adjustment of learning rate. IEEE Transactions on Neural Networks, **6**(1):157–169.

CUI, X., T. E. POTOK, and P. PALATHINGAL. 2005. Document clustering using particle swarm optimization. IEEE Swarm Intelligence Symposium, Pasadena, CA, pp. 185–191.

DAY, W. H. E. and H. EDELSBRUNNER. 1985. Investigation of proportional link linkage clustering methods. Journal of Classification, **2**:239–254.

GETOOR, L. 2003. Link mining: A new data mining challenge. ACM SIGKDD Explorations Newsletter, **5**(1):84–89.

GRIRA, N., M. CRUCIANU, and N. BOUJEMAA. 2005. Unsupervised and semi-supervised clustering: A brief survey. *In* Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval, Singapore, pp. 9–16.

GRUBER, T. R. 1993. Towards principles for the design of ontologies used for knowledge sharing. *In* Formal Ontology in Conceptual Analysis and Knowledge Representation. *Edited by* N. Guarino, and R. Poli. Kluwer: Deventer, the Netherlands.

HECHERMAN, D. 1995. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06.

HESS, E. J., N. KUSHMERICK. 2004. ASSAM: A tool for semiautomatically annotating Web services with semantic metadata. *In* The Semantic Web—ISWC 2004: Third International Semantic Web Conference, Vol. 3298 of Lecture Notes in Computer Science. *Edited by* S. A. McIlraith, D. Plexousakis, and F. van Harmelen. Springer: Hiroshima, Japan, pp. 320–334.

HOTHO, A., A. MAEDCHE, and S. STAAB. 2001. Ontology-based text clustering. *In* Proceedings of the IJCAI-2001Workshop of Text Learning: Beyond Supervision, Seattle, WA.

HOTHO, A., S. STAAB, and G. STUMME, 2003, Explaining text clustering results using semantic structures. *In* Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases: PKDD, vol. 2838 of LNAI. *Edited by* N. Lavraˇc, D. Gamberger, L. Todorovski, and H. Blockeel. Springer: Berlin Heidelberg, pp. 217–228.

Huang, X. and W. Lai. 2003. Identification of clusters in the web graph based on link topology. In Proceedings of the Seventh International Database Engineering and Applications Symposium (IDEAS'03), Hong Kong.

JAIN, A. K., M. N. MURTY, and P. J. FLYNN. 1999. Data clustering: A review. ACM Computing Surveys (CSUR), **31**(3):264–323.

GARETH, J., A. M. ROBERTSON, C. SANTIMETVIRUL, and P. WILLETT. 1995. Non-hierarchic document clustering using a genetic algorithm. Information Research, **1**(1).

KENNEDY, J., R. C. EBERHART, and Y. SHI. 2001. Swarm Intelligence. Morgan Kaufmann: New York.

KOLLER, D. and M. SAHAMI. 1997. Hierarchically classifying documents using very few words. *In* Proceedings of the 14th International Conference on Machine Learning (ICML), Nashville, TN, pp. 170–178.

KRISHNA, K. and M. MURTY. 1999. Genetic K-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, **29**(3):433–439.

LABROCHE, N., N. MONMARCHE′, and G. VENTURINI. 2003. AntClust, ant clustering and web usage mining. *In* Genetic and Evolutionary Computation Conference, Chicago, IL, pp. 25–36.

LARSEN, B. and C. AONE. 1999. Fast and effective text mining using linear-time document clustering. *In* Proceedings of SIGKDD'99, San Diego, CA, pp. 16–22.

LIKAS, A., N. VLASSIS, and J. VERBEEK. 2003. The global K-means clustering algorithm. Pattern Recognition, **36**(2):451–461.

MAEDCHE, A. and S. STAAB. 2001. Ontology learning for the semantic Web. IEEE Intelligent Systems, **16**(2):72–79.

MCQUEEN, J. 1967. Some methods for classification and analysis of multivariate observations. *In* Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, pp. 281–297.

MERWE, V. D. and A. P. ENGELBRECHT. 2003. Data clustering using particle swarm optimization. *In* Proceedings of IEEE Congress on Evolutionary Computation (CEC), Canberra, Australia, pp. 215–220.

MITCHELL, T. M. 1995. Machine Learning, Chapter 6. McGraw-Hill: New York.

OMRAN, M., A. SALMAN, and A. P. ENGELBRECHT. 2002. Image classification using particle swarm optimization. *In* Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), Singapore, pp. 370–374.

PATANE, G. and M. RUSSO. 2001. The enhanced-LBG algorithm. Neural Networks, **14**(9):1219–1237.

QUINLAN, R. J. 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann: San Francisco, CA.

RAGHAVAN, V. V. and K. BIRCHAND. 1979. A clustering strategy based on a formalism of the reproductive process in a natural system. *In* Proceedings of the Second International Conference on Information Storage and Retrieval, Dallas, TX, pp. 10–22.

SEBASTIANI, F., 2002. Machine learning in automated text categorization. ACM Computing Surveys, **34**(1):1–47.

STEINBACH, M., L. ERTOZ, and V. KUMAR. 2003. Challenges of clustering high dimensional data. *In* New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition doi: 10.1.1.99.7799.

STUMME, G., A. HOTHO, and B. BERENDT, 2001. Semantic web mining. *In* 12th European Conference on Machine Learning (ECML'01)/5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01), Freiburg, Germany.

STUMME, G., A. HOTHO, and B. BERENDT. 2006. Semantic web mining state of the art and future directions. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, **4**(2):124–143.

WANG, Y. and M. KITSUREGAWA. 2001. Link based clustering of web search results. *In* Second International Conference on Advances in Web-Age Information Management (WAIM), Xian, China, pp. 225–236.

WANG, Y. and M. KITSUREGAWA, 2002. On combining link and contents information for web page clustering. *In* Proceedings of the 13th International Conference on Database and Expert Systems Applications, Aix-en-Provence, France, pp. 902–913.

WITTEN, I. H. and E. FRANK. 2000. Data Mining, Practical Machine Learning Tools and Techniques, Chapter 6 (2nd ed.). Morgan Kaufmann: San Francisco, CA.

WOLPERT, D. H., and W. G. MACREADY. 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, **1**:67–82.

XING, E. P., NG, A. Y., JORDAN, M. I., and S. RUSSELL. 2003. Distance metric learning with application to clustering with side-information. *In* Neural Information Processing Systems, Vol. **15.** MIT Press: Cambridge, MA, pp. 505–512.

XU, R. and D. WUNSCH. 2005. Survey of clustering algorithms. IEEE Transactions on Neural Networks, **16**(3):645–678.

YAO, Z. and B. CHOI. 2003. Bidirectional hierarchical clustering for web mining. *In* Proceedings of the IEEE/WIC International Conference (Web Intelligence), Halifax, Canada, pp. 620–624.

YPMA, A. and T. HESKES. 2002. Categorization of Web pages and user clustering with mixtures of hidden markov models. Workshop Notes of the Fourth WEBKDD Web Mining for Usage Patterns and User Profiles at KDD'2002, Edmonton, Alberta, Canada, ACM, pp. 31–43.

ZAMIR, O. and O. ETZIONI. 1998. Web document clustering: A feasibility demonstration. *In* Proceedings of SIGIR' 98, Melbourne, Australia, pp. 46–54.

ZHANG, D. and W. S. LEE. 2004. Learning to integrate web taxonomies. Journal of Web Semantics, **2**(2):131–151.

ZHAO, Y. and G. KARYPIS. 2002. Evaluation of hierarchical clustering algorithms for document datasets. *In* Proceedings of the Eleventh International Conference on Information and Knowledge Management, CIKM'02, McLean, VA, pp. 515–524.

ZHAO, Y. and G. KARYPIS 2005. Hierarchical clustering algorithms for document datasets. Data Mining and Knowledge Discovery, **10**:141–168.